Web Based Active Server Pages Module for Engineering Students

Dr. Lisa Anneberg
Lawrence Technological University (LTU)
Southfield, MI 48075

Dr. Ece Yaprak
Wayne State University (WSU)
Detroit, MI 48202

Dr. Salman Talahmeh (PPU)
Palestine Polytechnic University
Hebron, West Bank

Abstract

A self-contained module introduces the engineering student to dynamic Web based computer architecture and programming. Active Server Pages (ASP) technology allows the designer to harness the interactive and dynamic nature of the Internet, and take advantage of its information and data. ASP combines HTML, Scripting Languages, and Components, which are familiar to students, and relatively easy to master. VBScript, in fact, has a 'basic' language derivative, and therefore is relatively quick for students to understand and pick up. This ASP module has been utilized for electrical engineering sophomore students, but could be readily adapted to other students for web-based applications as well. A big advantage of this Active Server pages module is that with a minimum of instructor and student effort, engineering students can have web-based programming exposure. Sophomore engineering students are encouraged to utilize the Internet, and now they have the proper experience.

ASP Definitions

ASP architecture is based on the client-server model. Active Server Pages is a programming environment that provides the ability to combine HTML, Scripting and Components to create powerful, dynamic, and interactive Internet applications.
The APS syntax includes the following items:

- Server side 'includes' (optional)
- HTML Code
- Script delimiters
- Script code
- Active X components (optional)
- ASP Objects (optional)

How ASP works

When a user browses an ASP-based site, the following occurs:

a. The user (client) requests an ASP page while browsing (a page that has the .asp extension), which is the second part of the URL:
   www.something.com/THISPAGE.asp
b. The browser requests the ASP page from the web server.
c. The web server executes all server-site scripts and produces the HTML page for the user/client.
d. The HTML page is sent to the user's browser.

Web services for Internet based computer architecture include the following:

1. Client browser - IE (Internet Explorer) or NS (Netscape)
2. Server software - IIS (Internet Information Server) or Apache (for UNIX)
3. Software architecture - Active Platform, including Active Server and Active Client including Active Server Pages
4. Languages - VB Script (client and server), JScript, JavaScript, PERL, REXX scripting languages and other supported languages.
5. Database of choice - SQL Server or MS Access

You can extend ASP scripts using COM components and XML. COM extends your scripting capabilities by providing a compact, reusable and secure means of gaining access to information. XML enables you to describe complex data structures by using a set of custom tags. If you are building a web application, you need to consider using both client and server scripting to make the application both dynamic and interactive.

The advantages to ASP include:
1. Leverages other Microsoft products / skills - such Access, Visual Basic, etc.
2. ASP is compiler free.
3. ASP protects proprietary logic.
4. Best of all, ASP does not have a steep learning curve.

The tools for developing ASP pages include: Notepad / Wordpad, Front Page, and InterDev. In this module, students begin with Notepad, since it is familiar. The code has HTML, VBScript, and JavaScript, which are familiar to the students. The various modifications for ASP are explained.

Getting Started with ASP

1. Once the code is typed in Notepad, the extension is named .asp, which is for ASP pages that sit on a server.
2. The delimiters of the code are similar to HTML:

a. Identify and start the scripting language:
   For server: *<%@ language = "scripting language" %>*
   For client: *<script language = "scripting language" >*

b. For inline server scripts that display an expression use the shorthand tags:
   <% =expression %>
c. For server function/procedure definitions, use:
   <Script language = "scripting language" runat = "server"> …… </script>
d. For client scripts, use *<script language = "scripting language"> ..... </script>*

Examples are derived in class, run on the laptop, and shown using an overhead projector. Students can see the development of programs, and the architecture becomes manageable. The students are encouraged to run the many examples, modify them, and extend them.

The following sets of examples are easily derived on the projected laptop, and executed so students may observe how ASP works.

1. HELLO WORLD EXAMPLE using server VB Script

This example displays the header "Exciting World of ASP Development!", displays the time of display, and displays seven lines of "Hello world!" in each line.

```
<%@ Language = VBScript %>
<HTML>
<HEAD>
<TITLE>Hello World, Active Server Page</TITLE>
</HEAD>
<BODY>

<H1><Font face=Arial size=6>Welcome to the
```

```
<FONT style="BACKGROUND-COLOR: #ffe4b5">Exciting</FONT>
World of ASP Development!</FONT></H1>

It is now <%= Time & " on " & FormatDateTime (date,vbLongDate) %>

<HR>

<%

Dim LineCount

For LineCount = 1 to 7
  Response.Write ("<FONT size=" & LineCount & _
  "> Hello World! </Font> <BR>")
Next

%>

<HR>

</BODY>
</HTML>
```

2. HELLO WORLD EXAMPLE using client JavaScript

This example shows an alternative way to do the above example, with a few differences that students can relate to. Different fonts, date function, and a while iteration loop are shown.

```
<HTML>
<HEAD>
<TITLE>Hello World, Active Server Page</TITLE>
</HEAD>
<BODY>

<H1><Font face=Arial size=6>Welcome to the
<FONT style="BACKGROUND-COLOR: #ffe4b5">Exciting</FONT>
World of ASP Development!</FONT></H1>
<BR>
<script language=JavaScript>
today = new Date()
document.write("Today\'s Date = " + today)
</script>

<HR>

<script language=JavaScript>
  var i = 1
  while (i < 8) {
  document.write("<BR><FONT size=" + i + ">" + "Hello World! </Font> <BR>")
     i++  }
</script>
<HR>
</BODY>
</HTML>
```

3. Third in-class example Using VBScript - Twist on Hello World
Again, this example illustrates various concepts that the students are already familiar with, in a distributed computing ASP environment.

```
<%@ Language=VBScript %>
<SCRIPT RUNAT=SERVER LANGUAGE=VBSCRIPT>
Sub SayHello ()
Response.Write("<H1>Hello! Today's date is " & Date & " </H1>")
End Sub
</SCRIPT>

<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<META HTTP-EQUIV="Content-Type" content="text/html; charset=iso-8859-1">
<TITLE>Syntax of ASP Applications</TITLE>
</HEAD>
<BODY bgcolor="#DBFFBF" link="#0000FF" vlink="#800080">
<!-- Insert HTML here -->
<%= "<H2>This line of text is displayed as output </H2>" %>
<%
Call SayHello ()
Response.Write ("<HR>")
%>

</BODY>
</HTML>
```

4. The following is interesting HTML code with forms and an ASP page. It is a relatively simple example, and is great for illustrating interesting ASP programming concepts and development:

a. HTML that includes a form:

```
<HTML>
<HEAD>
<TITLE>interaction with client information -- input screen</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<Form name = "frmTest" action = "frmHello.asp" method = “post”>
Name: <input type = “text” name = "txtName">
<input type = "submit" value = "send name">
</form>
</BODY>
</HTML>
```

b. ASP coding using forms, utilizing either VBScript or JavaScript:

```
<HTML>
<HEAD>
<TITLE>interaction with form -- response</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<%
dim strName     'the name the user entered on the form
strName = request.form ("txtName")
%>
```

```
Hello <% response.write(strName) %>, my friend.
</BODY>
</HTML>
```

5. The following code is a good twist on the above example. It is HTML code in ASP with Forms:

```
<HTML>
<HEAD>
<TITLE>Simple example -- input screen</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<b>Adding Machine</b>
<hr>
<Form name = "frmTest" action = "demo.asp" method = "post">
Enter Name: <input type = "text" name = "namer"><BR>
Enter Age: <input type = "text" name = "ager"><BR>
<input type = "submit" value = "Proceed">
</form>
</BODY>
</HTML>
```

5b. Another example using VBScript

```
<%@ language="VBScript" %>
<HTML>
<HEAD>

<script language="JavaScript">
function trial(data)
{
junker =   (data +  " here is some stuff done in secret")
document.write(junker)
}
</script>

<TITLE>simple example -- response</TITLE>
</HEAD>

<BODY BGCOLOR="#FFFFFF">
<form name="alisa">
<%
name = request.form ("namer")
age = request.form ("ager")
%>

Your name is: <%= name %><BR>
Your age is: <%= age %><BR>

<script language="JavaScript">
        trial("<%=name%>");
</script>

</form>
</BODY>
</HTML>
```

Many students have laptops, follow along with the program derivations, and get the hang of web based programs and programming. This skill is extremely useful for the electrical and computer engineer, and this rapid prototype methodology approach is very valuable.
Once the above in-class examples are derived, explained, run, and analyzed, the students have an assignment.
The initial Student Assignment is the following:

Develop an HTML form and an ASP that perform a hidden operation. Prompt for the name, password and purchase amount - then from your ASP, display the name, password, amount and discount:

Purchase <= 1000 then discount = 10%
Purchase > 1000 then discount = 20%

Conclusion

ASP, Active Server Pages programming, is a logical extension to HTML for the engineering student. ASP allows the engineer/programmer to take advantage of web-based applications by creating and controlling their design. Many engineering projects require web-based applications, and this introductory module is a set of hands-on projects designed to seamlessly introduce the topic and reinforce the concepts. Server and Client Scripting is important to maintain dynamic and interactive web pages. Considerations must be taken on using scripting to provide services to high percentage of users and high percentage of client browsers.

REFERENCES

1. Hettihewa, Sanjaya. (1999) Active Server Pages 2.0 in 21 days. Indianapolis, IN, SAMS, Inc..
2. www.microsoft.com/asp
3. Hatfield, Bill, Active Server Pagers for Dummies, Second Edition (1999), New York, NY, IGD Books, Inc.

Biographical Statements

LISA ANNEBERG
Dr. Anneberg has been an associate professor in electrical and computer engineering at Lawrence Technological University, in Southfield, MI since 1991. Her research interests include computer networks, error correction and detection, and freshman engineering design. She received the B.S. in industrial and operations engineering from University of Michigan in 1979, and the M.S. and Ph.D. in computer engineering from Wayne State University in 1983 and 1991, respectively.

ECE YAPRAK
Dr. Yaprak is an associate professor of engineering technology in the Division of Engineering Technology at Wayne State University in Detroit, MI. She is involved in several ongoing research projects, including field programmable chip laboratory development for NSF, distributed computing for the US Army and CBT for capstone design projects for NSF's Greenfield Coalition. She received the B.S. in electrical engineering from the University of Michigan in 1981 and the M.S. and Ph.D. in computer engineering from Wayne State University in 1983 and 1988, respectively.

SALMAN TALAHMEH
Dr. Talahmeh is an associate professor of computer engineering and Dean of Science and Technology at Palestine Polytechnic University in the West Bank. Dr. Talahmeh is involved in several industrial projects in web-based technology. Dr. Talahmeh received a B.S. degree in electrical engineering from Middle East Technical University in Turkey in 1982 and the M.S. and PhD. in computer engineering from Wayne State University in 1986 and 1996, respectively.